

多値論理回路に関する一設計法

藤田徹也 荒井 勇* 高松 衛* 中嶋芳雄*

要 旨

本稿では、2 値論理に関する出力多段NANDゲートの一設計法であるMA法を、多値論理系へ拡張する手法を示し、生成される初期回路の簡単化を含めた本方法のプログラム化と、種々の多値論理関数に対するシミュレーションの結果について述べる。

結果として、多値論理は2 値論理に比べて入力線数がより少なく設計できることが裏付けられた。また、簡単化の結果、初期回路は入力線数が77%、ゲート数が94%、段数が96%の規模まで縮小することができた。

キーワード

NANDゲート、MA法、多値論理系、簡単化、4 値論理関数

1 はじめに

近年のパーソナルコンピュータに搭載されるCPUなどの高集積回路では、論理演算の手法として2 値論理が広く用いられ定着した技術となっている。これに対し、2 値論理にはない利点を持つとされる理論が多値論理である。多値論理回路では、入力線数が2 値論理に比べ少なくなると言われている。

我々は、2 値論理回路をNANDゲートのみで設計する手法、MA法^[1]を発表してきた。複雑な高集積回路といえども、その回路を構成する要素はNANDゲートのような単純な素子から構成される。その中でNANDゲート(あるいはNOR)のみによる論理回路設計は、構造の一様性から集積回路化に適している。

多値論理においても、2 値論理の場合と同様に高集積化のための同一単一ゲートでの論理展開ができることが重要である。そこで本研究では、様々な多値論理系の中からAND、ORの他に基本演算子として x^a を考え、これに否定演算子 x^b を加えた多値論理を扱うSuらの多値論理系^[2]にMA法を拡張する手法を述べる。

る。

2 多値論理系

多値論理においては、2 値論理と異なり、AND、ORの他にこれまでに、種々の基本論理演算子が考えられ、様々な多値論理系とそれに見合う主項が定義されてきた。

その中で、AND、ORの他に基本演算子 x^a としてと否定演算子 x^b を定義したSuらの多値論理系にMA法を拡張する。多値論理系で否定演算が定義されていると、2 値論理のように、ド・モルガンの定理、 $\overline{A \cdot B} = \overline{A} + \overline{B}$ 及び、 $\overline{A + B} = \overline{A} \cdot \overline{B}$ が成り立つ。そのため多値論理系でもORゲートやANDゲートがNANDゲートのみで表せることになり、2 値論理回路をNANDゲートのみで実現するMA法が、多値論理系にも適用できると考えられる。

2.1 基本となる定義、及び定理

ここでは、まず多値論理系の基本となる定義、及び、定理について述べる。

2.1.1 準備

(定義1) 変数 X_1, X_2, \dots, X_n は、 p 値論理変数で $L=\{0, 1, 2, \dots, p-1\}$ の値をとる。 p 値 n 変数論理関数 $f(X_1, X_2, \dots, X_n)$ は、 L^n から L への写像である。図2.1のマップ(4値3変数論理関数 F の例)で与えることもできる。

		0				1				2				3			
X_1	X_2	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
	X_3	0	1	1	1	2	3	2	3	1	1	1	1	1	1	1	1
	X_4	0	1	1	1	1	3	3	3	3	1	1	1	1	1	1	1
	X_5	2	1	1	1	1	2	3	2	3	1	1	1	1	1	1	1

図2.1 4値3変数論理関数 F

完全系として、次の演算を考える。

(定義2)

OR演算 $X_i + X_j = \max(X_i, X_j)$ (1)

AND演算 $X_i \cdot X_j = \min(X_i, X_j)$ (2)

NOT演算 $\overline{X_i} = (p-1) - X_i$ (3)

(定義3)

$${}_a X^b = \begin{cases} p-1 & a = X = b \\ 0 & \text{その他} \end{cases} \quad (4)$$

$${}_a \overline{X^b} = \begin{cases} p-1 & X < a \text{ 又は } X > b \\ 0 & \text{その他} \end{cases} \quad (5)$$

但し、 $a, b \in L$ 、かつ、 $a \neq b$ である。 ${}_a X^b$ 、及び、 ${}_a \overline{X^b}$ をリテラルと呼び、冗長でないリテラルの積形式

$$X = {}_a X_1^{b_1} \cdot {}_a X_2^{b_2} \cdot \dots \cdot {}_a X_n^{b_n} \quad (6)$$

を項という。

(定理1) [4] n 変数の任意の多値論理関数 f は、積和形式

$$f = \bigvee_k (C_k \cdot {}_a X_1^{b_{k1}} \cdot {}_a X_2^{b_{k2}} \cdot \dots \cdot {}_a X_n^{b_{kn}}) \quad (7)$$

で表すことができる。

但し、 C_k, a_{ki} 、かつ、 b_{ki} は、 L の値をとる。

$$f = \bigvee_k (\overline{C_k} \cdot {}_a X_1^{b_{k1}} \cdot {}_a X_2^{b_{k2}} \cdot \dots \cdot {}_a X_n^{b_{kn}}) \quad (8)$$

の和積形式に変換できる。

但し、 ${}_a X_n^{b_{ki}} = {}_a X_n^{b_{ki}}$ 又は、 $\overline{{}_a X_n^{b_{ki}}}$ とする。

2.1.2 多値論理関数

例題として、図2.1の4値3変数関数 F を考えると、変数 X_1, X_2, X_3 は、4値論理変数で $L=\{0, 1, 2, 3\}$ の値をとる。4値3変数関数 $f(X_1, X_2, X_3)$ は、 L^3 から L への写像である。

積和形式で表すと

$$F = 1 + 2 \cdot {}_1 X_2^1 + 3 \cdot {}_1 X_1^1 \cdot {}_1 X_2^1 + 3 \cdot {}_1 X_1^3 \cdot {}_1 X_2^1 + 3 \cdot {}_1 X_2^1 \cdot {}_1 X_3^1 + 3 \cdot {}_1 X_2^1 \cdot {}_1 X_3^3 \quad (9)$$

となる。

又、図2.1の4値3変数関数 F の関数 \overline{F} をマップで表したものを図2.2に示す。

		0				1				2				3			
X_1	X_2	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
	X_3	0	2	2	2	1	0	1	0	2	2	2	2	2	2	2	2
	X_4	1	2	2	2	2	0	0	0	2	2	2	2	2	2	2	2
	X_5	2	2	2	2	2	1	0	1	0	2	2	2	2	2	2	2

図2.2 4値3変数関数 \overline{F}

図2.2のマップを積和形式で表すと、

$$\overline{F} = 2 \cdot {}_1 \overline{X_2^1} + 1 \cdot {}_0 X_1^0 \cdot {}_0 X_3^0 + 1 \cdot {}_0 X_1^0 \cdot {}_2 X_3^2 + 1 \cdot {}_2 X_1^2 \cdot {}_0 X_3^0 + 1 \cdot {}_2 X_1^2 \cdot {}_2 X_3^2 \quad (10)$$

となり、(定理1)から、

$$F = (1 + {}_1 X_2^1) \cdot (2 + {}_0 \overline{X_1^0} + {}_0 \overline{X_3^0}) \cdot (2 + {}_0 \overline{X_1^0} + {}_2 \overline{X_3^2}) \cdot (2 + {}_2 \overline{X_1^2} + {}_0 \overline{X_3^0}) \cdot (2 + {}_2 \overline{X_1^2} + {}_2 \overline{X_3^2}) \quad (11)$$

を導ける。

2.2 キューブ表現

p 値 n 変数関数 f は、幾何学的な表現として n 次元のキューブ(又は項) P^n に写像することができる。 P^n の頂点 (e_1, e_2, \dots, e_n) の n 個の組は、入力変数 (X_1, X_2, \dots, X_n) の n 個の組にそれぞれ対応している。 e_i は L の中の値をとる。各 e_i は、 p ビット $(b_0, b_1, \dots, b_{p-1})$ からなり、 e_i の $b_j(j=X_i)$ のビットが1、その他はビット0で表現される。

例えば、3値3変数関数 $F_1[L=\{0, 1, 2\}, V=\{X_1, X_2, X_3\}]$ において、キューブの頂

点が、3 入力の組 $(X_1, X_2, X_3) \in (2, 1, 0)$ である時、 $X_1=2$ であるから、 $e_1=(b_{10}, b_{11}, b_{12})$ の $b_{10}=b_{11}=0, b_{12}=1$ となり、 $e_1=(001)$ である。キューブ表現すると、 $e_1 e_2 e_3=(001)(010 \times 100)$ となる。

2.3 項表現とキューブ表現の関係

2.2 節で、項をキューブ表現する方法を示したが、リテラル $aX^b, \overline{aX^b}$ の積項をビット列で表現することになる。この変換のために、演算子 M をここで与える。

リテラルをビット列に変換する利点は、論理演算を行う上で有益である。特に多値 MA 法のアルゴリズムをプログラム化する際の、論理演算、及び許容項の表現等に最適な表現方法である。

2.3.1 ビット列演算子 M

リテラルをビット列で表現すると、変換演算子 M を用いて次のように示せる。

$$M(aX^b) = \underbrace{00\dots 011\dots 10\dots 0}_{b+1}$$

また、リテラルの否定 $\overline{aX^b}$ は、リテラルのビット列表現を反転したものとなる。

$$M(\overline{aX^b}) = \underbrace{11\dots 100\dots 01\dots 1}_{b+1}$$

例えば、4 値 3 変数関数で X, Y, Z は、4 値論理変数で $L=\{0, 1, 2, 3\}$ の値をとる時の以下のリテラルのビット列表現、及び、ビット列表現を用いた種々の演算を行うことができる。

$$\text{リテラル } {}^1X^3 \quad M({}^1X^3) = (0111)$$

$$\begin{aligned} \text{リテラルの否定 } \overline{{}^1X^3} \\ M(\overline{{}^1X^3}) = (1000) \end{aligned}$$

$$\begin{aligned} \text{項 } 2 \cdot {}^1X^3 \cdot {}^2Z^3 \quad M(2 \cdot {}^1X^3 \cdot {}^2Z^3) \\ = 2 \cdot M({}^1X^3 \cdot {}^0Y^3 \cdot {}^2Z^3) \\ = 2(0111 \times 1111 \times 0011) \end{aligned}$$

$$\text{AND 演算 } \overline{{}^0X^0} \cdot {}^3X^3 \quad M(\overline{{}^0X^0}) \cdot M({}^3X^3)$$

$$= (0111) \times (1110) = (0110) \quad {}^1X^2$$

$$\begin{aligned} \text{OR 演算 } {}^1X^2 + {}^3X^3 \quad + M({}^1X^2) + M({}^3X^3) \\ = (0110) \times (0001) = (0111) \quad {}^1X^3 \end{aligned}$$

3 2 値論理の MA 法

MA 法とは、2 値論理における一出力多段 NAND ゲート回路を生成する一設計法であり、肯定変数のみの積項で表される許容項を利用して論理関数の“1”の部分を実現していく。もしその許容項に“0”の部分を含めば、それを取り除く許容項を生成していき全部“1”、又は、全部“0”の許容項となるまで、この手法を繰り返して許容項の木を生成する。この許容項の木から一定の規則で NAND ゲート回路に置き換える回路設計法である。

MA 法は、積和形式で論理関数や許容項を扱っているが、ここからは分かりやすく関数 F をカルノー図などのマップで与えて説明する。

3.1 2 値論理の MA 法のアルゴリズム

2 値論理関数がマップで与えられているとする。マップの各セルは、2 進座標 (X_1, X_2, \dots, X_n) で表せるので、これを 10 進数に直して番号付ける。各セル毎に許容項が生成され、 n 変数のセル $j(X_{1j}, X_{2j}, \dots, X_{nj})$ に対して、各セルの座標成分をセル $k(X_{1k}, X_{2k}, \dots, X_{nk})$ とすると、セル j の座標成分に対して、座標成分がそれぞれ大きい、又は、等しいセル k の集まりを許容項と言う。

2 値論理の MA 法では、最初に木の根に相当する許容項を出力ゲート P_0 として、以下の方法で初期回路を生成する。

- (1) 真理値 1 のセルを、セル番号の小さい順に許容項 P_1, P_2, \dots, P_i を生成してすべて覆う。
- (2) (1) の許容項 $P_j (j=1, 2, \dots, i)$ について真理値 0 のセルを含むものがあれば、同様にセル番号の小さい順に真理値 0 のセルを取り除くように許容項を生成する。さらに、

(2)の許容項の中に、真理値 1 のセルがあれば、そのセルを開数上実現するため、ステップ(1)へ戻る(真理値 1 のセルを復活させる)。

この過程を許容項の中に真理値 1 と 0 が混在しなくなるまで続ける。こうして、できた許容項の木(図 3.1)から許容項をゲートに置き換え、許容項をなす変数をその入力リテラルとする。また、その許容項から生成した許容項からなるゲートの出力を入力として、多段NANDゲート回路(図 3.2)に置き換える。

2 値 4 変数関数 $F = X_1X_2 + \overline{X_1}X_4 + \overline{X_3}X_4$ を例に、MA法を適用して初期回路を生成する。許容項の木を図 3.1、そして、初期回路を図 3.2 に示す。

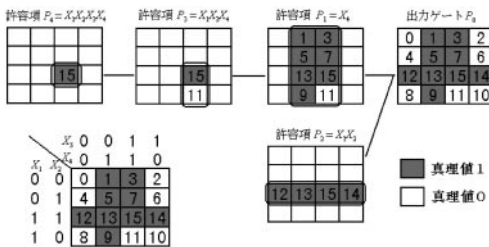


図 3.1 許容項の木

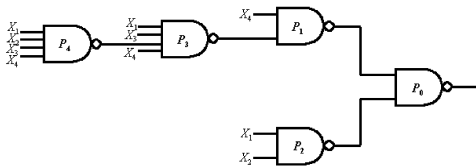


図 3.2 関数 F の初期回路

4 多値MA法

従来のMA法では、2 値論理を扱ってきたが、多値MA法では真理値が増加するので 2 値論理のMA法のアルゴリズムを改良する必要がある。

多値論理を扱う上で、従来の 2 値論理系とは別に多値論理系で、リテラル、項を表現し、また、論理演算を行わねばならない。そこで

本方法では、先に述べた否定演算の定義されたSuらの多値論理系にMA法を拡張する。

本章以降で示される回路図の中のNANDゲート素子は、多値NANDゲート素子である。

4.1 基本アルゴリズム

p 値 n 変数論理関数を f として、多値MA法の基本アルゴリズムを以下に示す。以下に“MA法”と記述してあるものは、2 値論理のMA法である。また“true”は、2 値論理における“1”であり、“false”は、“0”を表す。

- (1) 真理値 $k = p - 1$ から始める。
- (2) 真理値 k のセルを、trueのセル、 k より小さいセルをfalseのセル、そして、 k より大きいセルがあれば、そのセルをドント・ケアdのセルとしてMA法を適用する。
- (3) 真理値 k が $k = 1$ の場合は、ここで多値MA法を終了する。 $k > 1$ の場合は、 $k = k - 1$ としてステップ(2)に戻る。

ステップ(1)から(3)より、生成された k 毎の許容項の木を回路に変換するため、次に進む。

- (4) 真理値 k 毎に得られた許容項の木をそれぞれの回路へ変換する。各回路は、出力ゲートを持つが、回路の出力と真理値 k を入力とするNANDゲート(新たに真理値 k の回路の出力となる)を付加する。
- (5) k 毎の回路の出力ゲートを入力として持つ、NANDゲートですべての回路を 1 つにまとめる。この回路を多値MA法の初期回路とする。(図 4.1)

ドント・ケアdとは、真理値を“true”、“false”どちらとも考えられるという意味であり、許容項の生成個数を削減できる。すなわち初期回路のゲート数、入力線数を減らす効果がある。

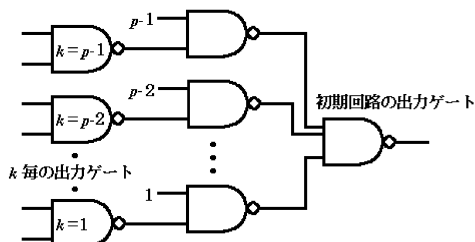


図4.1 真理値k毎の回路の合成(初期回路)

4.2 多値論理での許容項の基本概念

多値MA法では、2値論理のMA法と同様に、許容項を利用して許容項の木を生成し、初期回路を設計する。

p 値 n 変数関数 f で、この関数をマップで表すとセルの数は、 p^n 個となる。各セルは、 p 進座標 (X_1, X_2, \dots, X_n) で表されるので、これを10進数に変換して番号付ける。各セル毎に許容項が生成され、セル j ($X_{1j}, X_{2j}, \dots, X_{nj}$) に対して、各セルの座標成分をセル k ($X_{1k}, X_{2k}, \dots, X_{nk}$) とすると、

$$X_{1j} \quad X_{1k} \quad X_{2j} \quad X_{2k} \quad \dots \quad X_{nj} \quad X_{nk}$$

のセル k 、すなわちセル j の座標成分に対して、座標成分がそれぞれ大きい、又は、等しいセル k の集合を許容項と言う。また、許容項は、連続キューブとなる。

		0				1				2				3			
		X_2				X_2				X_2				X_2			
X_1	X_3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

セル3の許容項 $3X_3$

セル36の許容項 $2X_3 + 1X_3$

セル3の許容項 3X_3 セル36の許容項 $^3X_1^3 \quad ^1X_2^3$

図4.2 セルに対する許容項

4.3 部分マップを用いた許容項の生成

真理値 k ($k=0, 1, \dots, p-1$) が局部的に集中している関数(図4.3のマップ)の場合、trueの許容項を生成すると、多くのfalseのセルを含む許容項となることがある。そうなりとその許容項からfalseのセルを取り除くため、より多くのfalseの許容項を生成することになる。それは、すなわち初期回路のゲート数を増加

させることを意味する。

そこで、真理値 k 毎にtrueのセルがすべて存在する最小のマップ部分を生成し、その部分マップにMA法を適用すると、生成するtrueの許容項の中に含まれるfalseのセルの数を少なくできる。

許容項は、部分マップ内の各セルのみに対して、セル毎に生成する。また許容項は、連続キューブとなる。

ここで、例をあげて説明する。図4.3に局部的に真理値 $k=3$ が集中する4値3変数関数 F の $k=3$ に着目したマップを示す。 $k < 3$ (false)は、この図では空白としている。))

X_2	0				1				2				3			
X_3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
X_1	0						3	3								
	1						3	3	3							
	2						3	3	3							
	3						3	3	3	3						

図4.3 真理値 $k=3$ の4値3変数関数 F のマップ

このマップから、 $k=3$ のセルがすべて存在する最小のマップを生成すると、部分マップ $^1X_2^1 \quad ^1X_2^1 \quad ^1X_2^1 \quad ^1X_2^1$ (1111) (0100) (1111) を得る。この部分マップを図4.4に示す。

X_2	1			
X_3	0	1	2	3
X_1	0		3	3
	1	3	3	3
	2		3	3
	3	3	3	3

図4.4 部分マップ

4.4 多値MA法のアルゴリズム

多値MA法により初期回路を生成するアルゴリズムを説明する。

p 値 n 変数関数 F において X_1, X_2, \dots, X_n は、 p 値論理変数で $L=\{0, 1, 2, \dots, p-1\}$ の値をとる。真理値 k ($k=1, 2, \dots, p-1$) 毎にMA法を適用し、生成される許容項を P_{j-k} ($j=0, 1, 2, \dots, q$) とする。

ステップ(6)で用いるディスジョイント・シ

ョーブ(#)演算は、trueの許容項からfalseの許容項を取り除くための演算である。

ここで、真理値 k のセルをtrueのセル、真理値 k より小さいセルをfalseのセル、そして、真理値 k より大きいセルをドント・ケアdのセルとみなして、真理値 $k=p-1, p-2, \dots, 1$ の順に以下を行う。

(アルゴリズム 1)

- (1) 真理値 k に対して、部分マップ P_{M-k} を生成する。部分マップが生成できない場合、すなわち p 値 n 変数関数 F において、真理値 k が存在しない場合は終了する。
- (2) 部分マップ P_{M-k} に対して、ドント・ケアdを利用して部分マップの拡大を行う。拡大後の部分マップを新たに P_{M-k} とする。また、拡大できない場合は、そのまま部分マップ P_{M-k} を用いる。
- (3) 部分マップ P_{M-k} をfalseの許容項 P_{0-k} とする。
- (4) $i=0, j=0$ として、(アルゴリズム 2)に移る。

許容項 P_{i-k} ($i=0, 1, 2, \dots, s$)から、矛盾するセルを取り除くために生成する許容項を P_{j-k} とする。

関数 f は、 t を項として $f=t_1+t_2+\dots+t_m$ とする。

(アルゴリズム 2)

- (1) $j=j+1$ とする。
- (2) 関数 f をfalseの許容項 P_{i-k} とする。
- (3) 関数 f のマップ内に存在し、真理値がtrueでセル番号の最も小さいセルでtrueの許容項 P_{j-k} を生成する。関数 f のマップ内に真理値trueのセルが存在しない時、 $i=i+1$ としてステップ(2)に戻る。
- (4) 許容項 P_{j-k} に対して、ドント・ケアdを利用して許容項の拡大を行う。拡大後の許容項を新たに P_{j-k} とする。また、拡大できない場合は、そのまま許容項 P_{j-k} を採用する。
- (5) 生成済みの許容項の中で、許容項 P_{j-k}

と同一のものがある場合1つにまとめる(簡単化1)、 $i=i+1$ としてステップ(2)に戻る。

- (6) $f \# P_{j-k}$ の演算を行い、その結果を新たに関数 f とする。関数 f の時、ステップ(1)に戻る。また、関数 $f=$ の時、 $i=i+1$ としてステップ(1)に戻る。

(アルゴリズム 2)において関数 f とする許容項 P_{i-k} が、tureの許容項となる場合は、(アルゴリズム 2)の“true”を“false”と置き換えて行う。すなわち、trueの許容項からfalseのセルを取り除くためにfalseの許容項を生成し、falseの許容項からtrueのセルを取り除く(復活させる)ために、trueの許容項を生成する。

(アルゴリズム 2)は、 $i=j$ となるとき処理を終了し、次の真理値 $k=k-1$ として(アルゴリズム 1)に戻る。また、 k 毎に部分マップを生成し、許容項 P_{0-k} としているが、最後に許容項 $P_{0-k}=(1111 \chi 1111 \chi 1111)$ とする。

5 初期回路の簡単化

多値MA法で生成される初期回路(多段NANDゲート回路)は、非常に冗長な回路となる。そこで、初期回路の簡単化をする。

初期回路の簡単化として、(i)ゲート数の削減、(ii)ゲートへの入力線数の削減を行う。ゲート数の削減として、簡単化1、3、及び4、入力線数の削減として簡単化2を行う。

簡単化1は、多値MA法の中で行われる。簡単化2、3、4の行う順位によって簡単化後の回路は異なる。本方法では、入力線数の削減を優先的に行うので、簡単化1、2、3、4の順に行う。

ここで、ある許容項 P_{p-k} (真理値 $k=1, 2, \dots, p-1$)から矛盾するセルを取り除くために生成される許容項 P_{c-k} を P_{p-k} の子許容項と呼び、また、 P_{p-k} を P_{c-k} の親許容項と呼ぶ。

同様に、許容項の木をゲートに置換した回路でも、あるゲート P_{c-k} の出力がゲート P_{p-k} の入力とすると、 P_{p-k} を P_{c-k} の親ゲートと

ぜなら、段数が偶数段目のゲートは、trueの許容項からなり、奇数段目のゲートは、falseの許容項であるため、まとめることができないからである。

例えば、図5.3の奇数段目のゲート P_{2-3} と P_{2-2} は、入力変数がすべて同じで、子ゲートは共に持たないゲートなので同一ゲートとして1つにまとめる。

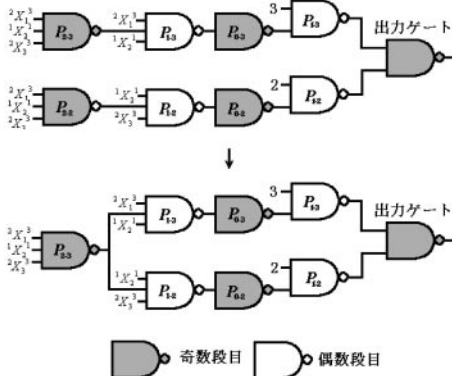


図5.3 簡単化3の削減例

5.4 積形式を利用したゲートの削減(簡単化4)

多値MA法で生成される初期回路は、一出力多段NANDゲート回路であり、その特徴としてド・モルガンの定理から奇数段目をORゲート、偶数段目をANDゲート、そして、奇数段目の末端ゲートをNOTゲートに置き換えることができる。 p 値 n 変数関数 F の初期回路において奇数段目の末端ゲート P_{e-k} (入力変数のみを持つゲート)の入力変数 (X_1, X_2, \dots, X_n) の積の否定を項 $\overline{X_{1e} \cdot X_{2e} \cdot \dots \cdot X_{ne}}$ とする。その親ゲート P_{p-k} は偶数段目となり、入力変数の積を項 $X_{1p} \cdot X_{2p} \cdot \dots \cdot X_{np}$ とすると、その2つの項の積

$X_{1p} \cdot X_{2p} \cdot \dots \cdot X_{np} (\overline{X_{1e} \cdot X_{2e} \cdot \dots \cdot X_{ne}})$ を演算した結果が1つの積項の場合のみ、変数をそれぞれ親ゲート P_{p-k} の入力変数とする。そして、ゲート P_{e-k} を削除する。それぞれの項は、ゲート P_{e-k} の場合、 P_{e-k} に対応する許容

項の否定を示し、ゲート P_{p-k} は、 P_{p-k} に対応する許容項である。また、親ゲートが複数ある場合は、すべての親ゲートと置き換えなければならない。

例として、図5.3の簡単化3を行った初期回路について簡単化4を適用する。

奇数段目の末端ゲートを検索するとゲート P_{2-3} が存在する。ゲート P_{2-3} に対応する許容項 $P_{2-3} = {}^2X_1^3 \cdot {}^1X_2^1 \cdot {}^2X_3^3 = (0011 \wedge 0100 \wedge 0011)$ である。ゲート P_{2-3} の親ゲートは、ゲート P_{1-3} と P_{1-2} である。各ゲートに対応する許容項は、許容項 $P_{1-3} = {}^2X_1^3 \cdot {}^1X_2^1 = (0011 \wedge 0100)$ (1111)と $P_{1-2} = {}^1X_2^1 \cdot {}^2X_3^3 = (1111 \wedge 0100)$ (0011)である。

許容項 P_{2-3} の否定と許容項 P_{1-3} の積をとると、

$${}^2X_1^3 \cdot {}^1X_2^1 (\overline{{}^2X_1^3 \cdot {}^1X_2^1 \cdot {}^2X_3^3}) = {}^2X_1^3 \cdot {}^1X_2^1 \cdot {}^0X_3^1$$

となる。

また、許容項 P_{2-3} の否定と許容項 P_{1-2} の積をとると

$${}^1X_2^1 \cdot {}^2X_3^3 (\overline{{}^2X_1^3 \cdot {}^1X_2^1 \cdot {}^2X_3^3}) = {}^0X_1^1 \cdot {}^1X_2^1 \cdot {}^2X_3^3$$

となる。

結果としてどちらも、1つの項となるのでそれぞれの親ゲートの入力変数と置き換える。そして、ゲート P_{2-3} を削除する。

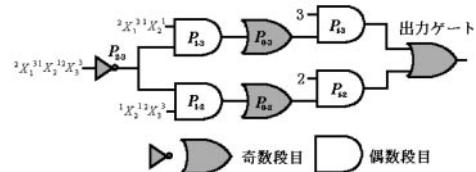


図5.4 置換されたAND-OR-NOT回路

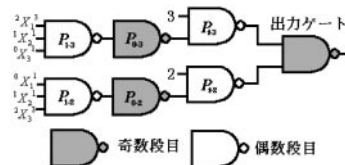


図5.5 簡単化4を適用後の初期回路

6 結果

多値論理に関する一設計法として、これまでに説明してきた多値論理系に拡張したMA法（多値MA法）及び、4つの簡単化のアルゴリズムを実際にC言語でプログラム化し、その実行した結果を述べる。

6.1 開発環境

プログラムの開発、及び実行に使用したコンピュータのスペックを以下に示す。

（CPU） Intel PentiumII 300MHz

（メモリ） 128MB

（OS） FreeBSD 3.2 - RELEASE

（コンパイラ） gcc

（プログラミング言語） C言語

6.2 実行結果

4値 n 変数関数($n=2\sim 7$)での本方法の簡単化の削減率、初期回路の回路規模の増加程度、及び、計算時間を示す。2値 n 変数関数($n=4, 6, 8, 10, 12$, 及び14)と真理値 $k=1$ のみの4値 n 変数関数($n=2, 3, 4, 5, 6$, 及び7)において初期回路の回路規模の比較を行った結果を述べる。

ここで、 p 値 n 変数関数をマップで表した場合のセルの総数は、 p^n 個である。これらのセルの中で、真理値0, 1, 2, ..., $p-1$ の値をとるセルの数の各割合が、 $c_0+c_1+c_2+\dots+c_{p-1}=1$ となるとき、この関数の真理表濃度を $c_0+c_1+c_2, \dots, c_{p-1}$ という。

6.2.1 4値 n 変数関数について

4値 n 変数関数において、2から7変数までの初期回路の入力線数(I)、ゲート数(G)、及び段数(S)の増加をみる。真理値表濃度は、各真理値0, 1, 2, 3に対して $c_0=c_1=c_2=c_3=0.25$ とする。同時に本方法で行う簡単化2、3、及び4でのゲート数(G)、入力線数(I)、及び段数(S)も表に示す。表は、先に述べた真理値表濃度になるようにランダムに発生させた

関数について、10回の実行結果の平均値である。

表6.1 多値MA法(簡単化1)

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	48.4	19.2	7.2
3	312.0	82.2	9.2
4	2088.6	380.2	11.2
5	12374.6	1708.4	11.2
6	70416.2	7535.2	11.4
7	376086.5	31882.2	12.5

表6.2 本方法(簡単化1、2)

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	41.8	19.2	7.2
3	239.2	82.2	9.2
4	1599.0	380.2	11.2
5	9572.6	1708.4	11.2
6	55402.0	7535.2	11.4
7	303617.2	31882.2	12.5

表6.3 本方法(簡単化1、2、3)

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	41.0	18.4	7.0
3	233.4	76.8	9.0
4	1583.8	369.8	11.0
5	9493.8	1654.4	11.0
6	55266.6	7461.8	11.4
7	303554.0	31862.5	12.5

表6.4 本方法(簡単化1、2、3、4)

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	38.2	16.8	6.4
3	219.4	71.2	8.4
4	1559.4	361.8	10.8
5	9424.4	1638.0	11.0
6	55215.4	7451.0	11.4
7	303532.2	31854.0	12.5

表6.1から、入力線数(I)、ゲート数(G)、そして、段数(S)が共に変数 n が大きくなるに従い増加している。

また、表6.2から表6.4を見ると、簡単化2、3、及び4で入力線数(I)、ゲート数(G)、

そして段数(S)がそれぞれ削減されていることが分かる。簡単化2は、入力線の削減のみを行い、ゲートの削減は行わない。簡単化3、4は、ゲートの削減を行い、それにもない入力線も若干削減している。

6.2.2 削減効率

6.2.1節の表6.2、6.3、6.4から入力線数(I)、ゲート数(G)、および段数(S)の削減効率を求める。多値MA法(簡単化1)のとき(表6.1)の、I、G、Sの値を基準として、簡単化(1、2)、簡単化(1、2、3)および簡単化(1、2、3、4)の各場合における値を残存率としてパーセンテージで表し、表6.5、6.6、6.7に示す。

表6.5 (簡単化1、2)での残存率 [%]

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	86.4	100.0	100.0
3	76.7	100.0	100.0
4	76.6	100.0	100.0
5	77.4	100.0	100.0
6	78.7	100.0	100.0
7	80.7	100.0	100.0

表6.6 (簡単化1、2、3)での残存率 [%]

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	84.7	95.8	97.2
3	74.8	93.4	97.8
4	73.7	97.3	98.2
5	76.7	96.8	98.2
6	78.5	99.0	100.0
7	80.7	99.9	100.0

表6.7 (簡単化1、2、3、4)での残存率 [%]

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	78.9	87.5	88.9
3	70.2	86.6	91.3
4	74.7	95.2	96.4
5	76.2	95.9	98.2
6	78.4	98.9	100.0
7	80.7	99.9	100.0

表6.5から表6.7を見ると、それぞれの割合にはばらつきがあるものの、本方法では、特に初期回路から入力線を20%ほど削減できている。また、ゲート数、段数は、平均して5%程度の削減にとどまり、変数が大きくなるほど削減率が悪くなっている。ゲートの削減個数は変数が大きくなるにしたがい増加するが、初期回路全体のゲート数が急激に大きくなるため、削減個数の割合は小さくなる。しかし、本方法の簡単化2、3、4によって、初期回路をより縮小することができた。

6.2.3 2値論理関数と4値論理関数の比較

多値論理回路は、2値論理回路に比べ回路の入力線数(I)を少なくできると言われている。ここでは、2値 n_2 変数関数($n_2=4, 6, 8, 10, 12$, 及び14)と真理値 $k=1$ のみの4値 n_4 変数関数($n_4=2, 3, 4, 5, 6$, 及び7)において、 $2^{n_2}=4^{n_4}$ のときの関数同士の入力線数(I)、ゲート数(G)、そして段数(S)を比較する。すなわち、関数をマップで表した場合のセルの総数が同じ関数同士を比べる。また、2値 n_2 変数関数、4値 n_4 変数関数双方の関数は、セル番号が同じセルの持つ真理値を同一にする。例えば、ある2値4変数関数のセル12=(1, 1, 0, 0)が真理値1であるとき、セルの総数が同じである4値2変数関数のセル12=(3, 0)を真理値1とするような関数同士を比較する。表は、真理値表濃度 $c_0=c_1=0.5$ として、先に述べた真理値表になるようにランダムに発生させた関数について、10回の実行結果の平均値である。

表6.8 2値 n_2 変数関数

変数の数(n_2)	入力線数(I)	ゲート数(G)	段数(S)
4	24.6	9.4	6.6
6	152.2	41.0	8.0
8	911.8	189.2	9.2
10	5654.8	859.6	9.8
12	31791.0	3796.8	10.2
14	165159.4	15785.0	11.4

表 6 .9 4 値 n_4 変数関数

変数の数(n_i)	入力線数(I)	ゲート数(G)	段数(S)
2	20.6	8.4	6.6
3	127.4	37.2	8.6
4	820.8	185.2	10.4
5	4880.2	838.6	12.2
6	24858.0	3705.4	12.2
7	135966.8	15493.4	13.0

表 6 .8 の 2 値 n_2 変数関数と、表 6 .9 の 4 値 n_4 変数関数を比較すると、本方法でも 2 値論理に比べ 4 値論理の方が入力線数(I)の小さい初期回路となっている。これは、多値 MA 法で生成される許容項が、2 値論理より 4 値論理の方が少ない変数の数の積項となるためである。すなわち許容項は、2 値 4 変数関数の場合、最大で 4 変数の積項で表されるのに対し、4 値 2 変数では、最大で 2 変数の積項となるからである。この許容項をゲートに置換すると 2 値 4 変数の場合は、1 つのゲートへの最大の入力変数の数は 4 入力であり、4 値 2 変数の場合は 2 入力となることから多値論理関数の方が、入力線数を少なくできると考えられる。また、ゲート数(G)も同様に 2 値論理に比べ 4 値論理の方が小さい初期回路となっている。よって、多値論理によって回路を設計する方が、規模の小さい回路を設計できると考えられる。

7 おわりに

本論文では、否定演算の定義された多値論理系へ MA 法を拡張する手法を述べた。すなわち、MA 法は本研究室で発表してきた 2 値論理に関する一出力多段 NAND ゲート回路の一設計法であり、これを多値論理に適用するために Suらの考案した多値論理系に拡張することができた。さらに、多値 MA 法で生成される初期回路を簡単化する手法も含めた本方法をプログラム化し、種々の多値論理関数を実行した。本研究では 4 つの簡単化を初期回路に対して、簡単化 1、2、3、そして 4 の

順に行った結果も示した。簡単化 1 を含む多値 MA 法で生成される初期回路を 100% とすると、その削減率は、入力線数(I)が 77%、ゲート数(G)が 94%、そして、段数(S)が 96% の規模まで初期回路を縮小することができた。

多段 NAND ゲート回路も 2 値論理回路と同様に 3 段化による初期回路の簡単化が考えられる。しかし、3 段化にすると出力ゲートなどの特定のゲートに入力線が集中し、多段 NAND ゲート回路の方が有効な回路と言える。

多値論理は、2 値論理に比べ入力線数(I)のより少ない回路を設計できると言われている。本方法でも 2 値論理と 4 値論理をある条件のもとで初期回路を生成し入力線数(I)を比較した。その結果、2 値論理よりも 4 値論理の方が入力線数(I)を少なく回路設計することができた。

今後、更に有効な簡単化の手法の開発が必要である。また、本方法では項表現で関数などを表しプログラム化したが、2 値論理では計算機のメモリ容量の縮小、及び、計算時間の短縮のため BDD(2 分決定木) が使われるようになった。そこで、多値 MA 法においても多値決定木によるプログラム化も課題として考える。

引用文献

- (1) 松田 秀雄, 宮腰 隆, 畠山 豊正: “多段 NAND ゲート回路の一設計法”, 情報処理学会研究報告設計自動化, DA - 67, 1993.
- (2) S. Suand P. T. Cheung: “Computer minimization of multivalued switching functions”, IEEE Trans. Comput., C-21, 9, pp. 995-1003, Sept, 1972.

参考文献

- [1] 松田 秀雄, 宮腰 隆: “部分マップ法による多値論理関数の主項導出について”, 電子通信学会論文誌, Vol.J68-D, No.6, Jun, 1985.

- [2] C.M.Allen and D.D.Givone : “ A minimization technique for multiple-valued logic systems ” ,
IEEE Trans. Comput., C-17, 2, pp.182-184,
Feb, 1968.
- [3] Z. G. Vranesic, E. S. Lee and K. C. Smith :
“ A many-valued algebra for switching systems ” ,
IEEE Trans. Comput., C-19, 10, pp. 964-971,
Oct, 1970.
- [4] 三根 久, 長谷川利治, 原田 尚文, 島田
良作 : “ 多線式多値論理回路網の一論理
設計法 ” , 信学論(D), 56-D, 3, pp. 186-193,
May, 1973.
- [5] 羽根 孝泰 : “ ドント・ケアを含む論理回
路をNANDゲート回路で実現する一設計法 ” ,
修士論文, TOYAMA-University, 1997.
- [6] 平松 徹也 : “ SBDD(共有二分決定グラフ)
を用いた多出力NANDゲート回路の設計法 ” ,
修士論文, TOYAMA-University, 1997.
- [7] 荒井 勇, 松田 秀雄, 中嶋 芳雄, 宮腰
隆 : “ 多値論理回路の一設計法 ” , 平成
11年度 . 電気関係学会北陸支部連合大会

A Method of Designing Multiple-Valued Logic Circuit

Tetsuya Fujita, Isam Arai, Mamoru Takamatsu, Yosio Nakajima

Abstract

In this paper, one method of extending MA method, which is one of designing 2-valued logic circuits using multiple NAND gates, to multiple-valued logic system is shown. The implementation of this method, including optimization of an initial circuit, and the result of simulation for some multiple-valued logic function is also discussed.

As a result, it becomes clear that the multiple-valued logic designing needs less input line comparing to a 2-valued logic. And, the optimization made an initial circuit more compact, 77% at input lines, 94% at gates, 96% at stages.

Keywords

NAND gates, MA method, Multiple-valued logic system, Simplification, 4-valued logic function